CodeWarrior for Microcontrollers V10.0

Device Initialization User Manual

Help version 2.12

Copyright 2011 Freescale Semiconductor, Inc.
PROCESSOR EXPERT is trademark of Freescale Semiconductor, Inc.

CONTENTS

1. Introduction 4

1.1. Features and Benefits 4
1.2. Basic Terms and Definitions 4
1.3. Quick Start 5
1.4. Rapid Application Development Tools 6

2. User Interface 8

2.1. Target CPU View 8
2.2. Inspector Dialog Box 10
2.3. Code Generation Options Dialog Box 12

3. Using the Tool 14

3.1. Peripheral Initialization Components 14
3.2. Code Generation And Usage 15
3.3. Defining Interrupt Service Routines 17
3.4. Changing The CPU 18
3.5. Converting Code Warrior Project To Use The Device Initialization 18

4. Help Revisions History 20

1. Introduction

Device Initialization is a fast, easy, and user-friendly way to configure and generate a CPU peripheral initialization code.

1.1. Features and Benefits

The key features of the Device Initialization tool are:

- Graphical user interface with CPU package, peripherals and pins.
- User-friendly access to the initialization setup of the CPU peripherals.
- Initialization code generator.
- User can select Assembly or C format for the generated code.
- Built-in detailed design specifications of the Freescale CPUs.
- Initialization options are automatically transformed into the peripheral control registers values.
- Easy to view control register values resulting from the parameter settings. Changes are immediately highlighted.
- Changes in the peripheral control registers values are transformed back into the component parameters.

The key benefits of the Device Initialization tool are:

- Easy to learn design environment.
- User friendly names of the peripheral features - no need to explore manuals for the control register details.
- Easy changes in initialization.
- Possibility to reuse the individual peripheral setup for other designs.
- No need to generate code to see the resulting peripheral control registers values.

1.2. Basic Terms and Definitions

Component - Peripheral Initialization component - is a component that encapsulates initialization of a peripheral. A component can be configured using the Inspector. See *3.1 Peripheral Initialization Components* for details.

CPU component - Component configuring the CPU core parameters. It cannot be removed from the project.

Design - All components (the CPU component and the Peripheral Initialization components) with their customized parameters.

Inspector - Window that allow to view and configure parameters of the selected component.

Internal peripherals - Internal peripherals of the MCU (ports, timers, A/D converters, etc. usually controlled by the CPU core using special registers).

ISR - Interrupt Service Routine - code which is called when an interrupt occurs.

Module - Source code module. Could be generated by the tool or created by the user and included in the project (user module).

Peripheral settings - Initialization parameters for an individual peripheral. These settings can be saved and restored from the file.

Popup menu - This menu is displayed when the right mouse button is pressed on some graphical object.

Properties - Parameters of the component. Property settings define the internal peripheral initialization state that will be converted to generated initialization code during the code generation process.

Target CPU - CPU derivative used in a given design.

1.3. Quick Start

This section describes how to create a simple design, configure a device, generate initialization code and use it in your application.

Step 1. Create an Empty Design

Use the Project Wizard to create a new project. It can be invoked by clicking on the **New Project Wizard** in the Startup screen or using the menu command **File > New > Project...**.

Follow the step by step settings and in the **Rapid Application Development Options** page and select the **Device Initialization** option.

Step 2. Configure Peripherals

The Target CPU window shows the CPU package with the available internal peripherals.

Click on a peripheral field to configure its initialization. A new peripheral initialization component is automatically created and the Inspector dialog window is displayed. It allows the user to view and change the parameters.

Use the Inspector dialog window that appears to setup initialization parameters of the peripheral as per your requirement and confirm it by clicking on the OK button. Use the same steps for all peripherals you wish to setup.

Step 3. Generate Code

Click the Generate Code button in the Target CPU window.

Within the Options dialog box, that appears immediately, **select the desired initialization code format: C or Assembly.**

Please note that on ColdFire CPUs, the assembly code generation is not available.

Within the Options dialog box, that appears immediately, you can specify the name of the output files you wish to generate (you can also keep a default name 'MCUInit') using the field *Generated module name*.

For more details on generated code, see the *3.2 Code Generation And Usage*.

Step 4. Use the Generated Code

The MCU_init initialization function call is automatically placed at the beginning of the main routine. Start writing your application code after this initialization function call.

1.4. Rapid Application Development Tools

Two tools are available in the CodeWarrior IDE for rapid application development: Processor Expert and Device Initialization. Both tools have many advanced features that lead to development cycle acceleration.

Features Comparison

Feature	Processor Expert	Device Initialization
Easy to use graphical IDE	✔	✔
Interactive design specifications of Freescale MCUs	✔	✔
Generated code	Peripheral Drivers	Initialization code only
Generated code languages	C	C or Assembly
Peripheral Init Components	✔	✔
Low Level Components	✔	✘
High Level Components	✔	✘
Project Configurations	✔	✘
User-friendly linker parameter file configuration	✔	✘
Generated code changes tracking	✔	✘
Timing settings in time units (such as seconds and bauds)	✔	✘
User components creation	✔	✘

Device Initialization

Device Initialization provides a fast and easy way to configure and generate an initialization source code for the CPU. It contains only one set of components: Peripheral Initialization Components. The code initializing the peripheral can be generated for use in Assembler or C.

Processor Expert

Processor Expert generates drivers in the C language that allows a hardware-independent access to MCU peripherals. It contains large library of components called Embedded Components. Embedded Components encapsulate the initialization and functionality of embedded systems basic elements, such as CPU core, CPU on-chip peripherals, standalone peripherals, virtual devices and pure software algorithms. These facilities are interfaced to the user using properties, methods, and events, such as objects in Object Oriented Programming (OOP).

Note: Processor Expert Embedded Components were formerly called Embedded Beans.

Description of the Embedded Components on different levels of abstraction:

- **High Level Components** - Basic set of components designed carefully to provide functionality to most of the microcontrollers in the market. An application built from these components can be easily ported to another microcontroller supported by the Processor Expert. This group of components can provide comfortable settings of a desired functionality such as time in ms or frequency in Hz, without the user knowing about the details of the hardware registers.
- **Low Level Components** - Components dependent on the peripheral structure that allow the user to benefit from the non-standard features of a peripheral. The level of portability is decreased due to a different component interface and the component is usually implemented only for a CPU family offering the appropriate peripheral. However, there is still possible to easily configure device's features and use a set of methods and events.
- **Peripheral Initialization Components** - Components on the lowest level of abstraction. An interface of such components is based on the set of the peripheral control registers. These components cover all features of the peripherals and are designed for initialization of these peripherals.

For more details, please see the Processor Expert documentation.

2. User Interface

Windows and Dialog Boxes

The Device Initialization user interface consists of the following windows that are integrated in the CodeWarrior IDE:

- **Target CPU** - Main window graphically showing CPU package, structure and components connected to the internal peripherals. It allows the user to easily to add components related to a specific peripheral to the design. See *2.1 Target CPU View* for details.
- **Inspector** - Window that allows the user to setup peripheral initialization components parameters. See *2.2 Inspector Dialog Box* for details.
- **Code Generation Options** - Dialog box with design and code generation related settings. See *2.3 Code Generation Options Dialog Box* for details.

2.1. Target CPU View

The Target CPU window is the main window of the Device Initialization design. The window contain the **control buttons** at the top of the window and **working area** that allows the user to browse and configure the CPU peripherals. To open this window, please use the pop-up menu for the CPU component in the project tree.

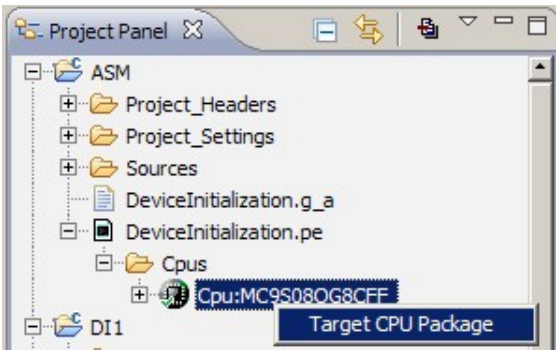


Figure 2.1 - CPU Component Pop-up Mmenu

Design Control Buttons

The window contains the following control buttons:

- **Select CPU package** - Lists available packages from the current target CPU. From the list of packages, the user can choose the one to be used in the design. See *3.4 Changing The CPU* for details.
- **Generate Code** - Invokes the Code Generation dialog allowing the user to generate the initialization code according to the current settings. See *3.2 Code Generation And Usage* for details.

Work Area

This area allows the user to configure the CPU peripherals by adding the Peripheral Initialization Components. See *3.1 Peripheral Initialization Components* for details.

The window shows:

- Peripherals available on the MCU and their allocation by components.
- Pins and their allocation by components.
- Useful information that is available in the status line if the mouse cursor is placed on pin, component or peripheral.

The following information about each pin is displayed on the package:

- Pin name (either default or user-defined).
- Icon of a component that uses (allocates) the pin.
- Direction of the pin (input, output, or input/output) symbolized by blue arrows, if a component is connected.

Hints

A hint appears when the mouse cursor is placed on a specific item:

A pin hint contains:

- pin number
- pin name
- owner of the pin (component that allocates it)
- short pin description

Note: The pin hint is available only in the package view mode

Component icon hint contains:

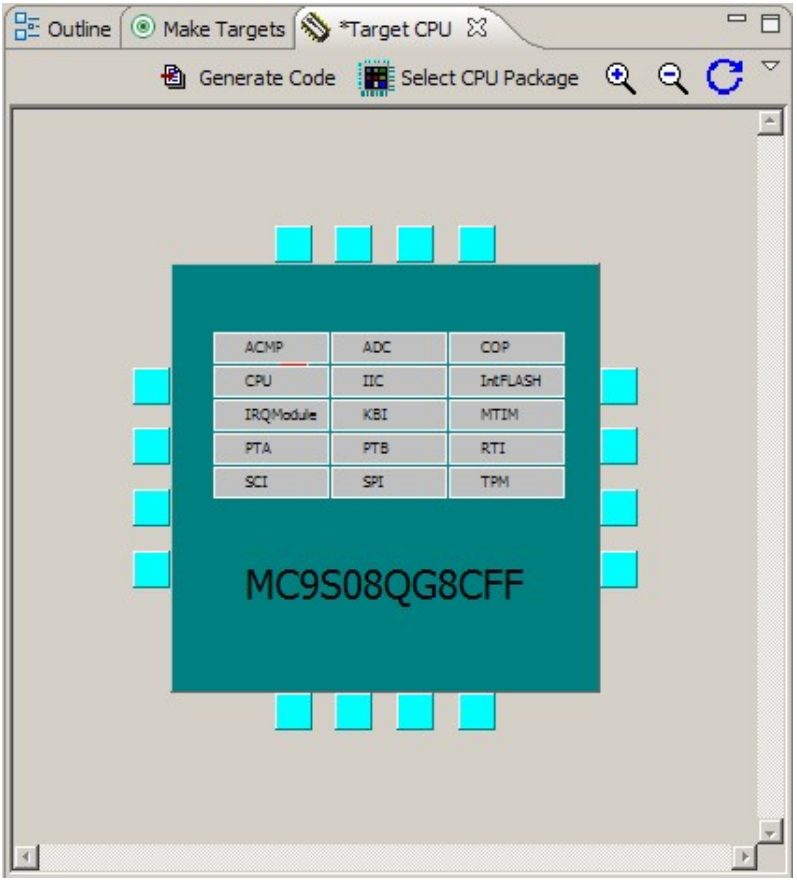
- component type
- component description

General Pop-up Menu

This menu appears when the user right-clicks anywhere in the Device Initialization window. It contains the following commands:

- **Zoom in** - Shows the help page.
- **Zoom out** - Shows the help page.
- **Rotate** - Shows the help page.

Sample Screenshot



2.2. Inspector Dialog Box

The Inspector dialog box allows the user to modify parameters of components configuring internal peripherals of the target CPU. (For more details on the peripheral components, please see *3.1 Peripheral Initialization Components*.)

Inspector dialog box consists of two panels:

- **Component Parameters** - Contains the parameters that influence the initialization of the selected device.
- **Register Details** - Contains the values of the individual control registers that is set by a generated initialization code.

Control Buttons

The buttons description from left to right:

- Disable Peripheral Initialization - Removes the currently opened peripheral initialization component from the design. This command is not available for the CPU component.
- - Opens the file selection dialog box and saves parameters to the selected file.
- - Opens the file selection dialog box and restores parameters from the selected file.

- - Saves the initialization. parameters and closes the window
- - Cancels changes and closes the window.

Component Parameters Panel

Component Parameters panel contains three columns:

- **Item names** - Items that are to be set are listed in the second column of the inspector. Groups of items describing certain features can be collapsed or expanded by double clicking on the first line of the group.
- **Selected settings** - Settings of the items are made in this column. If the user should select from pre-defined values, there is a drop-down menu with the list of available options. There is are radix change buttons 'H','D','B' for switching value radix to Hexadecimal, Decimal and Binary, where such change is reasonable.
- **Setting status** - Current setting or an error status may be reflected on the same line, in the rightmost column of the inspector. The error is also displayed in the item's hint.

Settings with errors are marked with red color and the description for the error can be found in the rightmost column.

A parameter can be presented as read-only and the user cannot change its content. Such read-only values are shown in gray.

Register Details

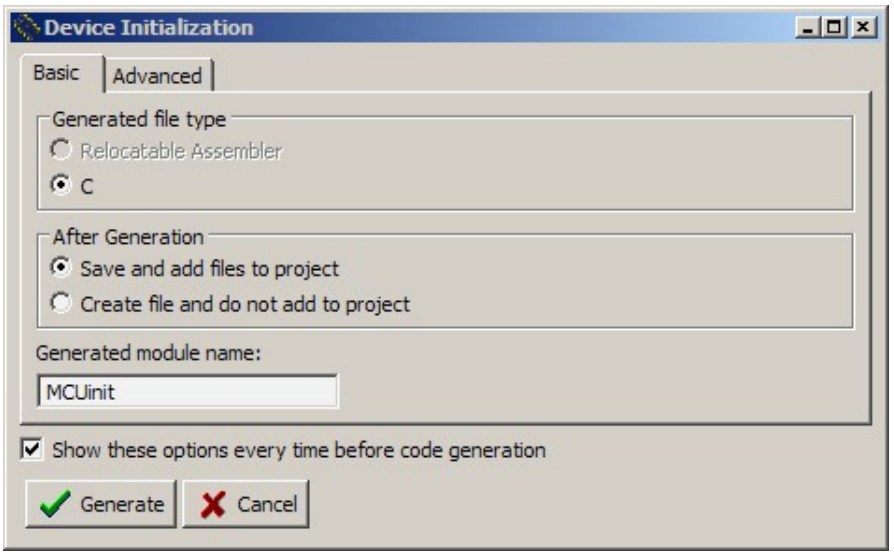
This panel shows values of individual control or data registers related to the currently selected CPU peripheral and reflects the settings in the *Component Parameters* panel.
On some peripherals there may be present an *Additionally modified registers* section within this panel that lists the registers that are initialized but belong to a different peripheral.
The following two types of rows can be found in this panel:

- **Whole register content**
  The row contains the following columns:
  - **Name** - Specifies name of the register according to the CPU datasheet.
  - **Init. Value** - Specifies the initialization value of a register or bit computed according to the settings in the *Component Parameters* panel. **This is the last value written by the initialization function to the register.**
    Note: For some registers, the value read from the register afterwards can be different than the last written value. For example, some interrupt flags are cleared by writing 1. Please see the MCU manual for details on registers behavior.

    If the row is active, the value can be changed using the keyboard. The field also contains a radix button (H,D,B) that allows to switch between Hexadecimal, Decimal and Binary formats of the value.
    The value that contains undefined bits is always shown in binary format.
  - **Address** - Specifies address of a register.
- **Individual bit of the register**
  These rows can be unfolded under each register contain values and the names of individual bits. Bits are sorted from the highest to lowest weight. The bit rows of a register can be shown or hidden using the plus/minus icons.

2.3. Code Generation Options Dialog Box

This dialog box is invoked at each code generation (using the **Generate Code** button). The user can specify the options influencing the format and placement of the generated code. Options are divided into two groups using tabs.



Basic Options

Basic options include the following options and option groups:

- **Generated file types** available:
  (For more information on generated files, please see the chapter *3.2 Code Generation And Usage*)
  - **Relocatable Assembler** - Generates the relocatable code in the assembly language. This option is not available for the absolute assembly projects.
  - **Absolute Assembler** - Generates an absolute code in assembly language. This option is supported only if it was selected in the CodeWarrior Project wizard.
  - **C** - Generates the code in the C language. This option is not available for the assembly projects.
- **After Generation**
  - **Save and add files to the project** - The files produced by Processor Expert are named using the value of the *Generated Module Name* field. The files are automatically added to the *Generated Code* folder of the Code Warrior project.
  - **Create file and do not add to project** - The code will be generated into the newly created untitled editor files.
- **Generated module name** - Specifies name of the files where the initialization code will be generated. If this field is empty, the '**MCUInit.c**' is used as default value instead.

Advanced Options

The following options modify the generation of code:

- **Generate register modification only if initialization value does not match reset state** - This option does not affect the registers writable only once (for example CONFIGx) nor the registers placed in FLASH (for example MORx).
- **Generate comments about register modification** - Source code will contain comments with descriptions of the registers values.
- **Generate interrupt vector table** - Interrupt vector table content will be generated to the output file.
- **Generate interrupt service routine templates** - Tool will generate an empty interrupt routines definitions for all enabled interrupts according to the components parameters. See *3.3 Defining Interrupt Service Routines* for details.
- **Generate initialization of registers writable only once** - These registers can be written only once after reset due to technological or safety reasons. This options enables the generation of initialization code for these registers.
- **Generate initialization of register placed in FLASH** - Initialization of these registers will be done during the programming of the application code to the FLASH memory.

Common Options

- **Show this dialog every time before code generation** - Using this check-box, the user can enable or disable appearance of this dialog box before every code generation.

3. Using the Tool

The sub-chapters describe basic principles and tasks related to device initialization.

- Peripheral Initialization Components
- Code Generation And Usage
- Defining Interrupt Service Routines
- Changing The CPU
- Converting Code Warrior Project To Use The Device Initialization

3.1. Peripheral Initialization Components

A Peripheral Initialization Component is an object that provides a human-readable interface to the initialization parameters of a selected on-chip CPU peripheral. Parameters of the Peripheral Initialization Component represent the settings of all peripheral registers in the clear tabular form. Names of the Peripheral Initialization Components are *Init_<peripheral>* and they are specific for each CPU family.

Adding Component

Components can be added to the design using the Target CPU window. Click on the unallocated peripheral to add a new component. The new component is preset to work with the selected device. Inspector dialog box appears allowing the user to configure parameters of the component.

Component Parameters and Device Registers

Every component contains a group of parameters (properties) that describe the desired initialization state of the device. These parameters are automatically transformed to values of control registers related to the selected peripheral. Inspector shows both - component parameters and resulting registers values.

Component parameters are grouped to several groups by type. The following groups are commonly present in peripheral initialization components:

- **Settings** - Common peripheral parameters.
- **Pins** - Configuration of the pins related to the peripheral.
- **Interrupts** - Configuration of the interrupts and interrupt service routines related to the peripheral. See *3.3 Defining Interrupt Service Routines* for details.
- **Initialization** - Parameters related to the peripheral initialization.

CPU components

A CPU component is the component that configures the parameters of the CPU core, such as clock source and frequency and power-saving capabilities. The CPU component is always present in design and cannot be removed. CPU component contains the following parameter groups:

- **Clock Settings** - Configuration of the CPU timing
- **Internal Peripherals** - Configurations of the peripherals not supported by separate components and settings that can be shared among components.
- **CPU Interrupts** - Configuration of the interrupts related to the CPU core.

Modifying Components Settings

Parameters of existing components can be configured using the Inspector dialog which can be opened by clicking on the component's icon in the Device Initialization window.

3.2. Code Generation And Usage

Starting The Code Generation

To generate code:

1. Click the *Generate Code* button in the Target CPU. The **Code Generation Options** dialog box with the code generation options appears. See *2.3 Code Generation Options Dialog Box* for details.
2. Click on the **Generate** button in the Options dialog box. The source code initializing the MCU registers according to the specified component parameters is generated. The generated modules are added into the project and can be found in the **Sources** sub-folder of the project.

Generated Code

During the Code Generation process the Device Initialization tool generates the initialization code for the CPU peripherals allocated by the components. The generated code reflects the configuration of the components' parameters and is contained in the function named *MCU_init*. The user should call this function at the beginning of the application code to initialize peripherals.

In the **ColdFireV1** MCUs, an additional function is generated, named *__initialize_hardware*. It contains settings of the clock source, the core registers and the system control registers (SOPTx, SPMSCx, all write-once registers). This function is called from the after-reset startup code.

The generated module consists of two files:

- **Implementation file** containing the code of the initialization function(s) and optionally the interrupt vectors table.
- **Interface file** containing the declarations that can be referenced by the user. This file is generated only if the files are stored to a disc (see below).

Depending on the *After generation* option, the files can be stored to a disk and added to the current project or just shown in the editor as untitled files. See *2.3 Code Generation Options Dialog Box* for details.

Device Initialization tool can generate the following types of initialization code:

- **Relocatable Assembler** - The implementation file has the extension **.asm** and the interface file has the extension **.inc**. This option is not available for the absolute assembly projects.
- **Absolute Assembler** - The implementation file has the extension **.inc** and must be included at the end of the user module, where address for code is selected (using ORG). Absolute assembler is supported only if it was selected in the CodeWarrior Project wizard.
- **C language** - The implementation file has the extension **.c** and the interface file has the extension **.h**.

A default name for the generated module is '*MCUInit*'. An Initialization code format, the generated module name, and other code generation options, can be configured in the Options dialog box. See *2.3 Code Generation Options Dialog Box* for details.

User Changes in the Generated Code

If the content of generated modules is written to the disk, it always overwrites the existing files during every code generation. As a result all the user modification are discarded with the following exceptions:

- user definitions (or generally any other source code) in .C (or .asm) placed in the special comment marks. In case of C language it looks like:
```
/* User declarations and definitions */
  User source code...
/* End of user declarations and definitions */
```
- content of interrupt service routines that are assigned to any interrupt(s) in the peripheral initialization components. See *3.3 Defining Interrupt Service Routines* for details.
- unused interrupt service routines (no component contains their name). They do not disappear but they are moved to the end of the file. The generated comments before and after the ISR must not be modified for this feature to work properly.

Note: No user changes in the .h file (or .inc in case of assembly language) are preserved. The file is always overwritten.

Using the Generated Code

To call MCU_init function from the main file, it's necessary to do the following modification in your code:

- Device initialization by default generates an interrupt vectors table containing all interrupt vectors (it can be disabled in the Options dialog box, see chapter *2.3 Code Generation Options Dialog Box* for details). Existing interrupt vector definitions have to be removed or commented out to avoid a conflict.

  Version specific information for RS08 family
  On the derivatives from the RS08 family the interrupt support is missing or is limited. See *3.3 Defining Interrupt Service Routines* for details.

  For details on configuring interrupt vectors in Device Initialization please see the chapter *3.3 Defining Interrupt Service Routines*.

- Add a call to the generated *MCU_init* function at the beginning of the application main routine. The newly created projects already contain this line.
  Note: The prototype or external declaration of the MCU_init function or a command including the interface file with this declaration should be present in the module where the MCU_init is called. In a new project, it is already there in the main file.

3.3. Defining Interrupt Service Routines

Some Peripheral Initialization components allow the initialization of an interrupt service routine. Interrupt(s) can be enabled in initialization code using appropriate parameters that can be found within the group *Interrupts*.
After enabling, the specification of an Interrupt Service Routine (ISR) name using the *ISR name* property is required. This name is generated to Interrupt Vector table during the code generation process. See *3.2 Code Generation And Usage* for details.
Please note that if the ISR name is filled it is generated into the Interrupt Vector Table even if the interrupt property is disabled.

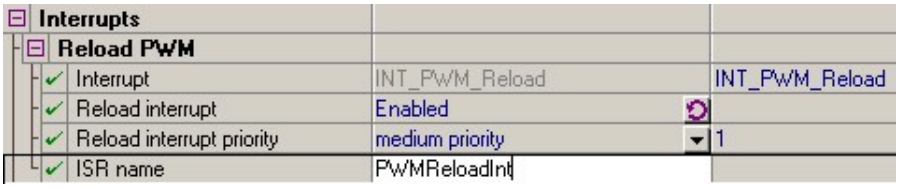


Figure 3.1 - Example Of The Interrupt Setup

Enabling or disabling peripheral interrupts during runtime has to be done by the user's code.

Version specific information for RS08 family
Derivatives of the RS08 family with CPU core version 2 support a single global interrupt vector. The interrupt doesn't support a vector table lookup mechanism as used on HC(S)08 devices. It is the responsibility of a routine servicing the global interrupt to poll the system interrupt pending registers (SIPx) to determine if an interrupt is pending. To support the single global interrupt vector there is a set of emulated interrupt vectors for each HW module defined. User can assign an interrupt service routine to these emulated interrupt vectors in a component. To emulate the interrupt vector table mechanism of the HCS08 family a global interrupt vector routine is generated, which performs check of the SIPx registers to determine if an interrupt is pending and calls appropriate interrupt service routines.

The order in which the SIPx registers are polled is affected by priority of the emulated interrupts. The priority can be in the range 0 .. number_of_interrupts-1 (e.g. 0 .. 15). Default priority value depends on the position of an associated bit in a SIPx register. The interrupt priority can be changed to any value within the allowed range. The lower is the number, the higher is the priority because the interrupts with a lower priority number are polled first. When two interrupts are assigned the same priority then the order in which they are polled depends on the default priority.

Interrupt Service Routines Code

The ISR with the specified name has to be declared according to the compiler conventions and fully implemented by the user. Declarations of the ISRs that do not exist yet can be generated automatically during the code generation process into the generated module if the option **Generate interrupt service routine templates** is enabled. See *2.3 Code Generation Options Dialog Box* for details.

Version specific information for RS08 family
All ISR on the RS08 family has to be defined as ordinary C functions (or assembly routines) because they are called from global interrupt handler that emulates individual interrupt functionality.

The contents of interrupt service routines, written by the user, that are assigned to any interrupt within the components parameters is protected against being overwritten during the code Generation process. In case the interrupt service routine is not assigned to any interrupt, it's moved to the end of the file.

Warning: *The user is responsible for synchronizing ISR names in the code and ISR names specified in components. If an ISR is renamed, the name has to be changed in the component(s) where this ISR name is assigned and vice versa. This has to be done before next code generation. Otherwise the newly specified ISR would not be found and the existing ISR with an old name will be treated as unassigned, that is, it will be moved to the end of file.*

3.4. Changing The CPU

Changing CPU package

The type of the CPU package can be changed using the *Select CPU Package* button in the Device Initialization window. See *2.1 Target CPU View* for details.

Switching the Project to a Different CPU Derivative

To switch to a different CPU derivative for the project, select **Project Change MCU / Connection** from the menu bar in the CodeWarrior IDE.

Components Assignment

If some peripherals of the MCU set by components are not supported by the new MCU derivative the project is switched to, a dialog box with a list of the unsupported items is shown. The user is asked to confirm that these items will be removed from the design.

3.5. Converting Code Warrior Project To Use The Device Initialization

This chapter guides the user through a conversion from a plain C or assembly (if assembly is supported) project to the project using the Device Initialization plugin and a peripheral initialization code generated by this tool.

Warning: Please note that in most cases this conversion involves necessary manual changes in the application code, because for example the register interrupt vectors table definitions created by the user often conflicts with Device Initialization definitions. Don't forget to backup the whole project before the conversion. Some files will have to be removed from the project. The conversion is recommended to experienced users only.

The following steps should be done to convert the project:

1. Select the menu command **File > New > Other...**
2. Within the "Select a wizard" dialog box select **Processor Expert / Enable Processor Expert for Existing C Project** and click on the **Next** button.
3. Select the project that you would like to convert and the project type **Device Initialization**.
4. Select the MCU that the project is designed for.
5. Select the MCU variant(s) and Processor Expert configuarions that you would like to have available in the project.

6. Review the actions that Processor Expert is about to perform. You can uncheck the checkboxes for items you would like not to be done. Please ensure you have backed-up your project before confirming before you confirm by clicking on Finish.
7. Add a call to the generated MCU_init function at the beginning of the application main routine. See *3.2 Code Generation And Usage* for details.
8. For Kinetis family projects, it's necessary to remove the files **kinetis_sysinit.c** and **kinetis_sysinit.h** from Project_Settings/Startup_Code. This module contains definitions that conflict with Processor Expert definitions.

4. Help Revisions History

The current revision number: **2.12** (Generated: 28.11.2011 15:51:29)

26.10.2011 Revision 2.12

- Added defalut value description to Code Generation Options section

24.5.2010 Revision 2.10
Minor corrections

INDEX

Adding Device Initialization 18
Changing CPU 18
Code generation 15
Component 4, 14
Converting project 18
CPU component 4
Creating ISRs 17
Design 4
Design steps 5
Features 4
Initialization code 15
Inspector 4, 10
Internal peripherals 4
Interrupt initialization 17
ISRS 17
Module 4
Options 12
Peripheral Initialization 14
Peripheral settings 4
Project creation 5
Properties 5
RAD tools 6
Target CPU 8, 5
Tools comparison 6
Using generated code 15